Bayesian Simulation

Aleksei Sorokin, asorokin@hawk.iit.edu, A20394300

4/1/2020

Load Data

```
df <- read.csv('data/all_coaches.csv')
data_dict <- list(
              N='Name',
              GR='Games Relative',
              WP='Win Loss Percentage',
              PGR='Playoff Games Relative',
              PWP='Playoff Win Percentage',
              CC='Conference Championships',
              C='Championships',
              HOF='Hall of Fame',
              S='Sport')
names(df) <- names(data_dict)
head(df)
```

```
##               N       GR        WP       PGR   PWP CC C HOF        S
## 1      AJ Hinch 6.308642 0.5580000 4.5454545 0.560  2 1   0 baseball
## 2   Aaron Boone 2.000000 0.6270000 1.2727273 0.500  0 0   0 baseball
## 3 Aaron Kromer 0.375000 0.3330000 0.0000000 0.000  0 0   0 football
## 4    Abe Gibron 2.625000 0.2740000 0.0000000 0.000  0 0   0 football
## 5     Adam Gase 4.000000 0.4690000 0.3333333 0.000  0 0   0 football
## 6    Adam Oates 1.585366 0.5752212 0.4375000 0.429  0 0   0   hockey
```

Split into train test datasets

```
set.seed(7)
train_frac <- 4/5
df_HOF_1 <- df[df$HOF==1,]
df_HOF_0 <- df[df$HOF==0,]
HOF_1_train <- sample(1:nrow(df_HOF_1),floor(nrow(df_HOF_1)*train_frac))
HOF_0_train <- sample(1:nrow(df_HOF_0),floor(nrow(df_HOF_0)*train_frac))
df_train <- rbind(df_HOF_1[HOF_1_train,],df_HOF_0[HOF_0_train,])
df_test <- rbind(df_HOF_1[-HOF_1_train,],df_HOF_0[-HOF_0_train,])
l1 <- sprintf('Train Fraction: %.2f',train_frac)
l2 <- sprintf('Hall of Fame Coaches: %d. (Train %d , Test %d)',
        nrow(df_HOF_1),length(HOF_1_train),nrow(df_HOF_1)-length(HOF_1_train))
l3 <- sprintf('Non Hall of Fame Coaches: %d. (Train %d , Test %d)',
        nrow(df_HOF_0),length(HOF_0_train),nrow(df_HOF_0)-length(HOF_0_train))
l4 <- sprintf('Overall: (Train %d , Test %d)',nrow(df_train),nrow(df_test))
```

```
cat(sprintf('%s\n%s\n%s\n%s\n',l1,l2,l3,l4))
```

```
## Train Fraction: 0.80
## Hall of Fame Coaches: 256. (Train 204 , Test 52)
## Non Hall of Fame Coaches: 1664. (Train 1331 , Test 333)
## Overall: (Train 1535 , Test 385)
```

Logistic Regression with only GR

using lm from R

```
model_1 <- glm(HOF ~ GR, data=df_train, family="binomial")
summary(model_1)
```

```
##
## Call:
## glm(formula = HOF ~ GR, family = "binomial", data = df_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3277  -0.4978  -0.4364  -0.4141   2.2491
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.44816    0.10628 -23.035   <2e-16 ***
## GR           0.11180    0.01188   9.414   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1203.0  on 1534  degrees of freedom
## Residual deviance: 1115.3  on 1533  degrees of freedom
## AIC: 1119.3
##
## Number of Fisher Scoring iterations: 4
```

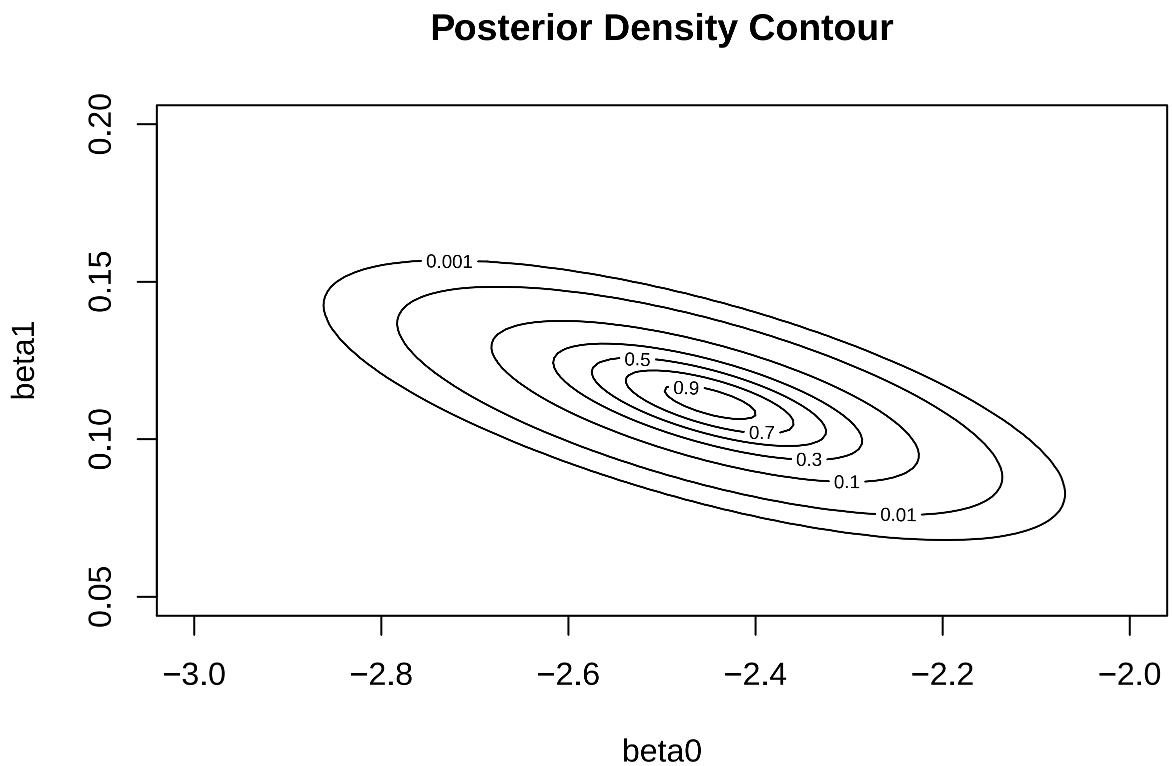
Posterior Simulation from non-informative prior

```
# constants
n_sims <- 1000
x <- df_train[['GR']]
y <- df_train[['HOF']]
length(x) == length(y)
```

```
## [1] TRUE
```

```
n <- length(y)
n_mesh <- 100
# compute posterior density on grid
beta0 <- seq(-3,-2,length=n_mesh)
beta1 <- seq(0.05,.2,length=n_mesh)
logl <- function(b0,b1){
  z <- b0+b1*x
  theta <- exp(z)/(1+exp(z))
```

```r
  d <- sum( y*z+log(1-theta) )
  return(d)}
log_post_dens <- matrix(NA, n_mesh, n_mesh)
for (i in 1:n_mesh){
  for (j in 1:n_mesh){
    log_post_dens[i,j] <- logl(beta0[i],beta1[j])}}
post_dens <- exp(log_post_dens - max(log_post_dens))
# contour plot
contour(beta0, beta1, post_dens, levels=c(.001,.01,.1,.3,.5,.7,.9),
        xlab='beta0', ylab='beta1', main='Posterior Density Contour')
```




```r
# simulations
beta0_density <- rowSums(post_dens)
beta0_idx <- sample (1:n_mesh, n_sims, replace=T, prob=beta0_density)
b0_sims <- beta0[beta0_idx]
b1_sims <- rep(NA,n_sims)
for (i in 1:n_sims){
  beta1_density_i <- post_dens[beta0_idx[i],]
  b1_sims[i] <- exp(sample(beta1, 1, prob=beta1_density_i))}
# intervals
I_b0 <- quantile(b0_sims,c(0.05,0.95))
I_b1 <- quantile(b1_sims,c(0.05,0.95))
# outputs
cat(sprintf('90%% interval estimate for a: (%.2f,%.2f)\n',I_b0[1],I_b0[2]))
```

```
## 90% interval estimate for a: (-2.64,-2.28)
```

```r
cat(sprintf('90%% interval estimate for b: (%.2f,%.2f)\n',I_b1[1],I_b1[2]))
```

```
## 90% interval estimate for b: (1.10,1.14)
```